Unlocking Efficiency: A Deep Dive into [Infrastructure as Code] (IaC)

Empowering Businesses with Automation, Consistency, and Cost Savings in Cloud Infrastructure Management

DEFIANCE DIGITAL

Table of Contents

INTRODUCTION 02

WHAT IS INFRASTRUCTURE AS CODE? 03

IAC EXPLAINED 04

HOW IAC CREATES EFFICIENCY 06

CODE OPTIMIZATIONS

- VPC Construct 09
- A More Redundant VPC 10
- Twingate Construct 11
- Log Ingestion Monitor for DataDog 12
- cdk8s Construct for External Secret Management 14
- A Microservice Project Definition Using Python 16

CONCLUSION 18

Introduction

Leveraging the power and flexibility of the cloud has become essential for organizations seeking scalability, agility, and cost efficiency. However, more than merely migrating to the cloud is required. If you migrate without implementing sustainable management practices and oversight, you expose yourself to the risks of overspending, outgrowing technology, and losing control over user experiences. Common mistakes include neglecting a regular assessment of your existing architecture and failing to implement supportive tools and tech infrastructure for your cloud.

Successful and sustainable cloud adoption requires not only implementation but consistent review and improvement to ensure great performance, resource utilization, and cost control. Cloud optimization is essentially fine-tuning various aspects of the cloud environment to ensure applications meet performance expectations, minimize unnecessary costs, and align cloud spending with strategic goals.
This guide offers a comprehensive checklist of cloud optimization best practices for companies seeking to provide competitive products while maintaining business sustainability. By following these practices, organizations can achieve a cloud environment with superior app performance and cost-effectiveness.

What is [Infrastructure as Code]?

As companies expand, they add complexity and challenges to maintaining their cloud infrastructure's integrity, security, and efficiency. One key example is when developers create software on cloud platforms like Amazon Web Services (AWS), Microsoft Azure, or Google Cloud, and they must manually configure the underlying infrastructure for each new application or system. This process becomes increasingly error-prone and time-consuming as the system grows.

Enter Infrastructure as Code (IaC), the superhero of the story. IaC automates the setup and management of infrastructure, ensuring consistency and reliability. It frees developers from the burdensome task of manual configuration, allowing them to focus on their core technology product.

This paper aims to provide technology leaders with an understanding of Infrastructure as Code and how Defiance Digital utilizes it to assist our clients. We will explore the fundamentals of IaC and its use cases and delve into the leading IaC-related cloud technology providers and tools, focusing specifically on Amazon Web Services. Additionally, we'll examine how a Managed Service Provider (MSP) can enhance the IaC process, including code examples.

IaC [Explained]

Traditional infrastructure management is like cooking a meal from a handwritten recipe. You gather ingredients and follow each step, but there's room for mistakes, like forgetting an ingredient or a step which can affect the final result.

In contrast, Infrastructure as Code (IaC) is like having a magical cooking machine. Instead of cooking manually, you provide the machine with the recipe card. Whenever you use that card, the machine prepares the meal exactly as intended, without missing a step. It's consistent, quick, and minimizes errors. This automation in cooking is similar to how IaC functions in technology setups.

During the early stages of software development or experimentation, software teams need to set up various technical environments, including shared development and quality assurance environments. On a small scale, manually configuring software resources is relatively simple. However, as the software matures and becomes critical, these setups must be consistently replicated for different purposes. Managing these setups involves many steps, details, and requirements, and relying on manual configurations becomes slow and prone to errors.

This is where Infrastructure as Code comes to the rescue. IaC is a method for managing and provisioning computing resources through machine-readable definition files rather than physical hardware configuration or interactive configuration tools. IaC uses plain text files, similar to special recipes, to define the necessary computer resources. These files are then interpreted by a provisioning engine, ensuring precise setup every time. The primary benefit of IaC is automation and consistency. By using code to represent infrastructure, teams can implement version control, collaborate more effectively, and automate the deployment and scaling of infrastructure. The provisioning engine is the magic cooking machine.

The Benefits

Using Infrastructure as Code (IaC) can offer several significant business benefits, including:

SPEED

IaC automates resource provisioning, significantly reducing setup and configuration time. This process enables quicker application deployment, expediting time-to-market.

COST SAVINGS

IaC optimizes resource management, preventing over-provisioning and minimizing manual intervention. This results in cost savings through resource efficiency and reduced labor expenses.

SCALABILITY & FLEXIBILITY

IaC facilitates easy scaling in response to changing workloads, ensuring infrastructure adapts dynamically.

REPRODUCIBILITY & DISASTER RECOVERY

Version-controlled IaC code allows for change tracking and rapid infrastructure restoration in case of failures or disasters.

IMPROVED COLLABORATION

IaC codifies and enforces security best practices during resource provisioning, bolstering infrastructure security, and compliance adherence. In addition, IaC results in risk reduction with things version-controlled.

SECURITY & COMPLIANCE

IaC codifies and enforces security best practices during resource provisioning, bolstering infrastructure security, and compliance adherence. In addition, IaC results in risk reduction with things version-controlled.

When to Use IaC

Many companies wait to consider implementing IaC until they reach a point where manual infrastructure setup is no longer efficient. However, delaying until this juncture can incur significant costs. A more strategic moment to contemplate IaC implementation is during the planning stages of expansion, the launch of a new product, or when it becomes evident that computational demands will experience rapid growth. Today's cloud vendors make it incredibly easy to provision new resources and scale almost infinitely and instantly, but the process is still complex, and errors at this phase can pose significant challenges to resolution. Developers and IT managers often find themselves taxed and operating inefficiently when dealing with errors and vulnerabilities prioritized over product releases.

Another best practice is to engage with a Managed Service Provider (MSP) when implementing IaC. While some may attempt to self-educate and establish an IaC process independently, the intricacies of IaC can make it challenging to create a comprehensive, robust, and maintainable system in the long run. You can streamline the process by collaborating with an MSP and outsourcing various aspects of IaC, ensuring optimal functionality and reliability.

In the upcoming sections, we will explore additional technologies related to IaC and provide examples of how Defiance Digital can efficiently and securely employ them to make IaC a success every time.

How IaC Creates [Efficiency]

IaC Tech & Tools

Various IaC tools and technologies are vital in defining, deploying, and maintaining infrastructure as code, making it more scalable, efficient, and repeatable. Some IaC tools are offered directly by cloud providers. Amazon Web Services provides CloudFormation, which encompasses both the specification language (JSON/YAML files) and the provisioning engine.

That being said, IaC is often agnostic to specific cloud providers. While some IaC tools are associated with a particular cloud (e.g., AWS CloudFormation), others like Terraform are multi-cloud and allow you to manage resources across various platforms. This means you can maintain a consistent approach to infrastructure management even if you use multiple cloud providers. Other systems use IaC as a management technique such as DataDog and Kubernetes. Defiance Digital also uses frameworks like the CDK for Terraform, and cdk8s, to help manage IaC across many cloud platforms.

Below is a glossary of some common IaC terms and technologies:

- **AWS CloudFormation: A service provided by AWS for defining and provisioning AWS infrastructure resources using JSON or YAML templates.**
- **AWS Cloud Development Kit (CDK): An open-source development framework that allows developers to use familiar programming languages to define and manage AWS resources using code.**
- **Terraform: A popular IaC tool using declarative language to define infrastructure as code. It supports multiple cloud providers and can manage various resource types.**
- **Azure Resource Manager (ARM) Templates: You can define Azure infrastructure as code using JSON templates for resource provisioning and management.**
- **Google Cloud Deployment Manager: Like CloudFormation and Terraform, it enables IaC on the Google Cloud Platform.**

- **Ansible, Puppet, and Chef: Configuration management tools that can include IaC capabilities by defining infrastructure and application configurations.**
- **Docker and Kubernetes: Containerization technologies that provide a consistent application environment and can be managed as code for deployment and scaling.**
- **Jenkins and CircleCI: Continuous Integration/Continuous Deployment (CI/CD) tools that can be used to automate the deployment of infrastructure changes defined in IaC scripts**
- **Version Control Systems (e.g., Git): These are crucial for tracking changes to infrastructure code, enabling collaboration, and ensuring versioning of configurations.**
- **Configuration Management Databases (CMDBs): Tools like Consul and etcd can be used to store and manage configuration data for IaC scripts.**

AWS CDK

AWS has developed the CDK to introduce an abstraction layer over CloudFormation's template language, utilizing imperative, general-purpose languages such as Typescript, Python, Java, C#, and Go. The AWS CDK construct system promotes code reuse, reduces redundancy, and offers flexibility by allowing extensive logic integration into IaC. Moreover, it is easily testable through standard tooling.

Since its inception in 2019, the AWS CDK has gained significant traction in the AWS community. Defiance Digital has been an early adopter and continues to leverage its advantages. We have found that other construct-based projects like CDK for Terraform, cdk8s, and projen have similarly delivered substantial Total Cost of Ownership (TCO) savings.

USE CASE

Code Optimization 1

VPC Construct

The code snippet below shows a stack named VpcStack, a reusable definition of AWS resources for a private network with outbound internet access; it has one construct– the standard Vpc construct. This code snippet represents what Defiance Digital would write in the CDK to generate a new set of resources within AWS. The snippet has six lines of code, but without our CDK usage here, it would take approximately 670 lines of code to do with CloudFormation natively. This is a massive saving in development and maintenance time, dramatically reducing the code ownership in a system, which means faster development time, lower maintenance costs, and a lower TCO.

```
1
2 export class VpcStack extends Stack {
3  constructor(scope?: Construct, id?: string, props:
4 StackProps = {}) {
5   super(scope, id, props);
6   new Vpc(this, 'Vpc');
7  }
8 }
```

The Vpc construct also has parameters that drive what it defines in AWS resources. Parameters like maxAzs which controls how many availability zones the VPC is deployed to. It also provides a natGateways parameter to manage costs versus redundancy. There are over a dozen parameters an engineer can provide to customize how their particular VPC is created.

USE CASE

Code Optimization 2

A More Redundant VPC

As seen in the example below, Defiance Digital would take the maxAzs property and the natGateways and set them to 6. This creates a highly redundant Vpc configuration, dramatically simplifying the code from an engineering perspective. A total of 1250 lines of CloudFormation code are generated for the engineer. Without the CDK, an engineer would have to spend time building and maintaining a series of templates that meet various combinations of reliability and cost savings concerns. That series of templates would have a considerable total cost of ownership without the CDK

```
export class VpcStack extends Stack {
 constructor(scope: Construct, id: string, props:
StackProps = {}) {
  super(scope, id, props);
  new Vpc(this, 'Vpc', {
   natGateways: 6,
   maxAzs: 6,
  });
 }
}
```

USE CASE

Code Optimization 3

Twingate Construct

Another advantage of working with an MSP for IaC is access to additional tools and products and the know-how to use them, such as the access management tool Twingate. At Defiance Digital, we leverage Twingate to provide zero-trust network access to VPCs in AWS. To ease the deployment, we have created a unique construct that we can easily drop into any CDK codebase to quickly deploy Twingate connectors to an environment.

```
export class TwingateConnectorsStack extends Stack {
 constructor(scope: Construct, id: string, props:
TwingateConnectorsStackProps) {
  super(scope, id, props);
  for (const i = 0; i < props.numberOfConnectors; i++) {
   new TwingateConnector(this, `TwingateConnector-${i}`);
  }
 }
}
```

USE CASE

Code Optimization 4

Log Ingestion Monitor for DataDog

The CDK for Terraform brings constructs to Terraform, allowing engineers to define Terraform resources using familiar building blocks and allowing Defiance Digital to create reusable Terraform code customizable for each client's needs.

One example is the DataDog monitor in the code snippet below, which helps manage log ingestion thresholds. The code below sets a limit on daily log ingestion and triggers an alert if the limit is exceeded, which allows our Datadog clients to control their system costs.

```
export class LogIngestionMonitor extends Monitor {
  constructor(scope: Construct, id: string, config:
LogIngestionMonitorConfig) {
    const notificationTargets = config.notificationTargets
      ? `Notify: ${config.notificationTargets}`
      : "";
    const criticalThreshold = config.criticalThreshold ?? "5000000.0";
    const criticalRecovery = String(Number(criticalThreshold) * 0.6);

    let logIngestionMessage;
    if (!config.message) {
      logIngestionMessage = `${notificationTargets}\n\nThe number of daily
ingested events is high.`;
    } else {
      logIngestionMessage = config.message;
    }

    super(scope, id, {
      ...config,
      name: config.name ?? "Datadog Log Usage",
      type: "query alert",
      query:
        config.query ??

`sum(last_1d):sum:datadog.estimated_usage.logs.ingested_events{*}.as_coun
t() > ${criticalThreshold}`,
      message: logIngestionMessage,
      monitorThresholds: {
        criticalRecovery: criticalRecovery,
        critical: criticalThreshold,
      },
      newHostDelay: config.newHostDelay ?? 300,
    });
  }
}
```

USE CASE

Code Optimization 5

cdk8s Construct for External Secret Management

Some of Defiance Digital's customers use Kubernetes as a platform, running Elastic Kubernetes Service on AWS. To define the Kubernetes resources, Defiance Digital uses cdk8s, creating multiple constructs for the client to use for their microservice architecture. This reduces TCO by keeping logic centralized and making it easy to create new microservice applications.

The following is an example of Defiance Digital creating a reusable cdk8s construct representing a kubernetes resource for syncing secrets between AWS Secrets Manager and a kubernetes cluster secret.

```typescript
/**
 * A construct which represents a Kubernetes ExternalSecret.
 * See <https://external-secrets.io/> for more information.
 */
export class ExternalSecret extends ApiObject {
  constructor(scope: Construct, id: string, props: ExternalSecretProps) {
    super(scope, id, {
      apiVersion: 'external-secrets.io/v1beta1',
      kind: 'ExternalSecret',
      metadata: {
        name: props.secretName,
      },
      spec: {
        dataFrom: [
          {
            extract: {
              conversionStrategy: 'Default',
              decodingStrategy: 'None',
              key: props.awsSecretKey,
            },
          },
        ],
        refreshInterval: props.refreshInterval || '1h',
        secretStoreRef: {
          kind: 'SecretStore',
          name: 'secretsmanager-secret-store',
        },
        target: {
          creationPolicy: 'Owner',
          deletionPolicy: 'Retain',
          name: props.secretName,
        },
      },
    });
  }
}
```

USE CASE

Code Optimization 6

A Microservice Project Definition Using Python

Platform resources aren't the only things that need management– code in Git repositories also do. Many teams share common repository files like GitHub actions, pull request templates, and package configurations. Traditionally, an engineer did this work through a templating engine like Yeoman or by cloning a boilerplate repository. However, these methods fall short since the file is created initially and left to the individual repositories and teams to maintain. Changes can be applied manually, which leaves a broken upgrade path, resulting in high management costs as time goes on.

Projen encapsulates and shares these standard components across multiple repositories in 'projects'. With projen, the files can be upgraded over time as requirements change. Upgrading individual repositories requires an update of the project dependency and a simple command line interface (CLI) program to update to the latest version of the project definition. This can even be automated through more centralized GitHub Actions defined in the projects.

The code snippet below highlights how Defiance Digital uses Projen to centralize important components like Docker compose files, GitHub action files, baseline CDK projects, and pull request templates across multiple Git repository projects.

```
export class PythonProject extends python.PythonProject {
  branchWorkflow?: BranchGithubWorkflow;
  trunkWorkflow?: TrunkGithubWorkflow;

  constructor(options: PythonProjectOptions) {
    const branch = options.trunkBranch ?? defaultTrunkBranch;
    const contents = '...';
    super({
      ...options,
      projenrcPython: false,
      githubOptions: {
        pullRequestLint: false,
      },
      sample: false,
      readme: { contents },
      moduleName: options.name,
      version: options.version ?? '0.1.0',
      authorName: options.authorName ?? 'Defiance Devops',
      authorEmail: options.authorEmail ?? 'devops@defiance.ai',
    });
  //  ...  removed for brevity:
  // create docker-compose file
  // create GitHub action for branch
  // create GitHub action for trunk
  // create baseline cdk8s project in subdirectory
    // create pull request template
    // add common gitignores
  }
}
```

[Conclusion]

Infrastructure as Code (IaC) emerges as the superhero solution for modern businesses grappling with the complexities of cloud infrastructure. It bestows organizations with the invaluable gifts of consistency, reliability, and efficiency that are game changers in managing business-critical technology. However, even though IaC is table stakes for technology-forward companies today, that doesn't mean it's easy. Even with access to the right tools and resources, the TCO required to operate IaC successfully is tremendously high. Remember this as your organization grows and moves to optimize your technology management. Focusing on products is a priority for growing businesses, and it is advantageous to outsource the more technical and exhaustive technology requirements to service providers with the capacity to handle it flawlessly.

About [Defiance]

Founded in 2020 out of Defiance Ventures, Defiance Digital is an AWS managed services provider offering pay-as-you-grow cloud services and consulting for small and medium businesses. We focus on delivering personalized support and exceptional results through direct access to elite cloud engineers who embrace our 'customers as co-workers' ethos. Our mission is to maximize cloud benefits while minimizing complexity and costs, allowing our clients to focus on their core business.

Our team of cloud experts offers end-to-end support, from strategy to execution, providing our clients with reliable, secure, and scalable solutions tailored to their unique needs. We foster strong relationships with AWS, Datadog, Lacework, Clumio, and other strategic partners to provide the best-of-breed security, observability, automation, and public cloud solutions. We operate with transparency, thoughtfulness, proactivity, and agility and constantly evolve to remain valuable partners for our scaling customers.

We'll Take It From Here

WEBSITE

DEFIANCEDIGITAL.COM

EMAIL

SALES@DEFIANCEDIGITAL.COM

ADDRESS

201 S COLLEGE ST - 1590,
CHARLOTTE, NC 28202